I see something similar happening in AI chip companies that I do in self driving cars. You have many wildly overcapitalized companies that have 0 product market fit, never mind revenue or profit.

Not a single one of these companies has a product I can buy! The current leader in AI chips is unquestionably NVIDIA, and they have a card I can buy. So that leaves the question, are the companies banking on a stupid business model, or do they just not have anything? I tend to think it's the latter. Remember: Unless it's a from company with a track record like Apple, secrecy is just cover for "we don't have shit"

Startups (without chips you can buy)

Graphcore ($682M)

Here is a Technical Report on the Graphcore IPU Architecture. It looks insanely hard to program for. And while they have a buy now button, it links to a contact us form. If a company has a contact us form gating sales, it means that their product isn't actually competitive in the market if you did the research yourself, but they are hoping a sales guy will wrongly convince you otherwise.

While I had to get to the 7th page of the report to find the problem, it's the same mistake as the Cell SPE. They rely on a many core chip with a custom weird communications protocol. You want to try to maintain fast code for that? Game devs hated the SPE, why wouldn't ML devs hate this?

Groq ($362M)

They seem to get the idea right, or at least match what I would do with a single big core running at around 1 GHz. Here is their Technical Report. I have heard that they get the details wrong though, and the fact that they are still adversing their ResNet-50 performance (a 2015 era network) speaks to that.

The problem is obvious on the first page of report, the same mistake as Itanium, "But this approach places a heavy burden on the compiler, which must comprehend the twin pulses of instruction flow and data flow while optimizing function-unit utilization. The compiler must schedule all data

movement, manage the memory and function units, and even manually fetch instructions." This push it to static software strategy has never worked in the history of computer architecture.

I'll leave with one more sentence from the report, "The memory units also store VLIW instructions, which are 2,304 (144x16) bytes wide" For reference, RISC-V instructions are 4 bytes wide. Hope you like fitting 7 instructions in your 16kB icache!

Tenstorrent ($234M)

Their chip is called Grayskull, and there's almost no information published on this architechture, and I was told the only way to get a demo would be to sign something promising to not say anything bad about it. That doesn't inspire much (read: any) confidence in it being good, I'm nice if people are honest and transparent, less nice if they aren't. But now that Jim Keller is there, perhaps there is hope.

Cerebras ($112M)

WE MAKE BIG CHIP! CHIP REALLY BIG! BIG MUST BE GOOD BECAUSE BIG! BIG CHIP MEAN BIG INVESTMENT! Good luck controlling 400,000 cores in any sane way.

Companies (with deployed chips for training)

NVIDIA

The king of AI chips. The 3090 isn't even a bad deal! The problem is the 5x overpriced A100, and the facts that, one, they have a monopoly, and two, the chips are still mainly designed as gaming GPUs with AI as an afterthought. While the 432 Tensor Cores have an insane number of FLOPS (8x4x4 = 128 FMAs each), it's almost impossible to keep them fed from memory, or even cache, and in reality performance is 10x worse. (this is where we get our 10x speed from)

Google

The TPU is based around TensorFlow. While it does work with PyTorch, it is really meant to have things compiled and laid out in RAM in specific ways. This is the "push it to software" pattern that I see fail over and over, while it

may work for an organization like Google, it's not practical for a normal organization trying to do ML. Hence why Google doesn't sell the TPUs, they sell services that can use them in their cloud. Nobody wants custom weird services that lock you in to Google, and this is why they are losing to AWS and Azure in the cloud space.

The best documentation I could find is here, along with this presentation on the changes from v2 to v3. It seems they have very large 2D arrays, 128x128 (reduced in v2 from 256x256 in v1). And VLIW instructions (at least only 322-bit), pushing all the complexity into a compiler.

The TPUv4 docs are out, and they look okay. Wish I could buy one. They also are using ML to do layout for the TPUv5.

AMD

Similar downsides to NVIDIA, but with much less investment in the software side. I don't know anyone who uses these to train. From twitch chat: "ROCm = please use kernel version 5.4.1.4.12.44.1 and only 5.4.1.4.12.44.1" aka unusable in practice.

Huawei

The Huawei Ascend910 made it on the list for MLPerf 0.7, so I included it here, even though I can't buy the chips or use them it seems. They seem to use a 16x16x16 matrix unit, not a bad choice. But it only supports FP16, which is a software hassle for training. Found more info here, it seems similarish to the Apple Neural Engine.

Habana/Intel

I can supposedly rent or maybe even buy the Habana Gaudi, though I don't actually see how to in practice. It's a 16nm chip, with 8 "Tensor Processing Cores", a 16-bit integer "GEMM engine", and 4 ports of HBM2 RAM. It's not on MLPerf, so it's likely not working or very uncompetitive.

UPDATE (10/30/21): I no longer like this plan. After talking with a few people, I realize I misunderstood where the power draw came from. The biggest cost is in moving the bits. While I still think it's very possible to beat NVIDIA as discussed below, a simple RISC-V accelerator add on won't do it. We may have finally gotten to the point where VLIW makes sense.

What I would build (**Cherry Computer**):

Our goal would be to beat NVIDIA and build training chips that are 10x faster at training (because we have the cache bandwidth to feed our tensor cores) and cost 10x less (because we aren't overcharging and can omit all the GPU silicon). Same D2C business model, just with purpose built hardware for ML training instead of shoehorning it into a gaming device.

I've been working on an extension to RISC-V that includes a 32x32x32 matrix multiply unit, and a few other wide vector style instructions. I would support only the TF32 datatype (19 bits), so it would well with training out of the box without gradient scaling (which is really a ton of software complexity to hack the larger floating point range into FP16). While training with smaller datatypes is possible, it's super annoying!

A lesson that seems to be forgotten over and over again is that VLIW doesn't work (Itanium) and fancy multicore doesn't work (Cell SPE). You have to make a chip people love to program in hardware! Otherwise, you push it to software and it never happens. How is that Itanium compiler coming along?

I've been learning computer architecture lately, I wrote a RISC-V core in a day. I think the idea is a single RISC-V core with a really wide decode path and good out of order support. The M1 has a 630 element reorder buffer, we won't even need this much to keep our MACs fed. While the argument against this sort of architecture is that it uses power, this is only a problem in massive multicore designs! If your basic instruction is a 32x32x32 matmul (65k FLOPS), this will be dwarfed by the power used by the FMACs. I don't think the branch predictor needs to be that good either, ML code is straightforward loops, and it's identical every run through.

One of the biggest problems with non NVIDIA training solutions is terrible software support. This isn't because it's hard to write, it's because it's hard to write with performance for the weird architecture chips these companies came up with. Since this chip is a single core, wide decode path superscalar, writing the code should be very easy, even to support all of PyTorch with decent performance.

Cherry One: My first step would be to write open source Verilog implementing this core that can run in the S7t-VG6, the Xilinx Alveo, or the FK33. According to my initial benchmarks, assuming 1 instruction per cycle and 1 GHz clock speed, this would comfortably outperform a 3090. This FPGA card costs $7500, so it wouldn't be cost effective to use, but this is the